# Test coverage para Laravel

verificar si xdebug esta instalado

```
php -v
```

Instalar xdebug

https://xdebug.org/download

renombrar el archivo en

```
C:\xampp\php\ext\php_xdebug.dll
```

en el php.ini añadir estas líneas

```
[xdebug]
zend_extension = xdebug
xdebug.remote_autostart = 1
xdebug.profiler append = 0
xdebug.profiler_enable = 0
xdebug.profiler enable trigger = 0
xdebug.profiler_output_dir = "c:\xampp\tmp";
xdebug.profiler_output_name = "cachegrind.out.%t-%s"
xdebug.remote enable = 1
xdebug.remote_handler = "dbgp"
xdebug.remote host = "127.0.0.1"
xdebug.remote log="c:\xampp\tmp\xdebug.txt"
xdebug.remote_port = 9000
xdebug.trace_output_dir = "c:\xampp\tmp";
xdebug.remote_cookie_expire_time = 36000
xdebug.mode=coverage
xdebug.start_with_request=yes
```

# **Generar Reporte de Cobertura**

Para generar un informe de cobertura de código, usa PHPUnit con el indicador --coverage.

# Generar un informe en consola:

```
Copiar código
php artisan test --coverage
```

### Resultado

```
      Console\Kernel
      16 / 66.7%

      Exceptions\Handler
      160.0%

      Httpt/Controllers\Appi\JournalController
      160.0%

      Http/Controllers\Controller
      160.0%

      Http/Controllers\JournalController
      160.0%

      Http/Middleware\Nathenticate
      160.0%

      Http/Middleware\RedirectIncryptCookies
      160.0%

      Http/Middleware\PerentRequest SDuringMaintenance
      160.0%

      Http/Middleware\Prestrimstrings
      160.0%

      Http/Middleware\Prestrimstrings
      160.0%

      Http/Middleware\TrustHosts
      0.0%

      Http/Middleware\Varisthosts
      160.0%

      Http/Middleware\Validatesignature
      160.0%

      Http/Middleware\VarifyCsrfToken
      160.0%

      Models\Journal
      160.0%

      Models\Journal
      160.0%

      Models\User
      160.0%

      Providers\AppServiceProvider
      160.0%

      Providers\AppServiceProvider
      160.0%

      Providers\RoadcastServiceProvider
      160.0%

      Providers\RoadcastServiceProvider
      160.0%

      Providers\RoadcastServiceProvider
      160.0%

      Providers\RoadcastServiceProvider
      160.0%

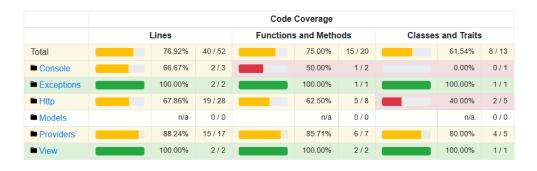
      Providers\RoadcastServiceProvider
```

# Generar un informe en HTML:

```
Copiar código
php artisan test --coverage-html=coverage-report
```

Los resultados estarán disponibles en la carpeta coverage-report.

## Resultado



### Legend

Low: 0% to 50% Medium: 50% to 90% High: 90% to 100%

Generated by php-code-coverage 10.1.16 using PHP 8.1.12 and PHPUnit 10.5.38 at Mon Nov 18 17:02:42 UTC 2024.

# Midiendo el type coverage con PEST en Laravel Raúl López https://raullg.com

En primer lugar, tenemos que instalar Pest y su extensión para el type coverage:

### **Para instalar Pest:**

composer require pestphp/pest --dev --with-all-dependencies

./vendor/bin/pest --init

### Para instalar el plugin:

composer require pestphp/pest-plugin-type-coverage --dev

Una vez instalado todo, podemos medir el type coverage de nuestro código sin escribir ni un solo test a través del siguiente comando:

```
./vendor/bin/pest --type-coverage
```

Esto nos dará un output tal que así:

```
        app/SourceCode/BitbucketProvider.php
        100%

        app/SourceCode/Contracts/AccountInfoProvider.php
        100%

        app/SourceCode/Contracts/DownloadsZipFile.php
        100%

        app/SourceCode/Contracts/ReceivesZipFile.php
        100%

        app/SourceCode/Contracts/SourceCodeProvider.php
        100%

        app/SourceCode/Contracts/ShandlesWebhook.php
        100%

        app/SourceCode/Contracts/RegistersWebhook.php
        100%

        app/SourceCode/Costartst/RegistersWebhook.php
        100%

        app/SourceCode/Cistarts/Provider.php
        100%

        app/SourceCode/Cistarts/Provider.php
        100%

        app/SourceCode/Costarts/Provider.php
        100%

        app/SourceCode/Costarts/Provider.php
        100%

        app/Console/Commands/CreateDemoRepositories.php
        100%

        app/Console/Commands/GenerateProjectDocumentation.php
        100%

        app/Console/Kernel.php
        100%

        app/Console/Kernel.php
        100%
```

Una captura del type coverage de CodexAtlas

Además, configurando esto como un paso en nuestro pipeline, podemos **requerir** un mínimo para nuestro type coverage, y así asegurarnos de que cualquier actualización de código tenga que estar tipado al máximo posible:

```
./vendor/bin/pest --type-coverage --min=70
```

En caso de no pasar el type coverage, obtendríamos algo así:

```
ERROR Type coverage below expected: 94.6%. Minimum: 100.0%
```

Esto, al ejecutarse, devolverá un código de respuesta distinto de cero, que será interpretado por las líneas de comando como un error y no permitirá pasar el pipeline.